- Draft -

Dynamic 3D Maps and Their Texture-Based Design

Jürgen Döllner, Klaus Hinrichs
Institut für Informatik, University of Münster
{dollner, khh}@uni-muenster.de

Abstract

Three-dimensional maps are fundamental tools for presenting, exploring, and manipulating geo data. This paper describes multiresolution concepts for 3D maps and their texture-based design. In our approach, 3D maps are based on a hybrid, multiresolution terrain model composed of data sets having different topological structure, for example a coarse regular grid combined with by triangulated microstructures. Any number of texture layers can be associated with the terrain model. For each texture layer, the multiresolution structure builds a texture tree which is linked to geometry patches of the multiresolution terrain model. The terrain model together with multiple texture layers can be rendered in real-time, in particular if multitexturing is available. Texture layers can be combined by high-level operations such as blending and masking, and can be rebuilt at run-time. This mechanism simplifies the implementation of visual exploration tools and of procedural, automated map designs. 3D maps facilitate the visual simulation of environmental issues in spatial support systems, virtual reality applications, real-time GIS, and interactive cartography.

1. Introduction

In many applications of interactive computer graphics, 3D maps are important tools for presenting, exploring, and manipulating geo data. In medium and large scales, dense graphical elements and a high level of abstraction are often required to communicate spatio-temporal information about environmental objects, their relationships, and natural phenomena. Hence, 3D maps can be used as planing instruments by experts as well as presentation tools for non-experts [20]. 3D maps which can be designed and configured in real-time enable interactive visual tools for data analysis and decision making. These *dynamic 3D maps* achieve a vivid and terse representation of spatio-temporal data. As a consequence, a more intuitive access to spatio-temporal data as well as an improved communication between geo-data and users result [18].

This paper describes concepts for implementing dynamic 3D maps based on a hybrid, multiresolution digital terrain model and multiple texture layers.

The multiresolution terrain model used in this approach permits to integrate geometry data having different topological structure – and can therefore precisely model morphologically important terrain parts. Typically, a terrain model is composed of a coarse regular grid combined with TINs representing microstructures of the terrain such as river beds or streets.

The multiresolution terrain model can be associated with any number of texture layers. For each texture layer, a multiresolution texture data structure is generated; the selection of an LOD texture patch depends on the geometric resolution of the associated terrain patch. To render a textured 3D map, the terrain patches actually used determine relevant LOD texture patches. Multitexturing, now available even on "low-cost" graphics hardware, enables an efficient rendering of multiple texture layers.

Multiple texture layers are deployed for 3D maps in various ways. (1) Typically, a texture layer visualizes thematic data related to a 3D map at different levels of detail. (2) With high-level layer operations such as blending and masking, texture layers can be combined in a single 3D map. (3) A texture layer may also be rebuilt and redesigned at run-time. This mechanism permits the implementation of dynamically changing 3D maps: the 3D map can choose appropriate visual designs according to design rules. These rules may consider camera settings and screen resolution. Therefore, we can automate the design of the map contents. (4) Furthermore for visualizing thematic data the map design may use representations different from textures such as 3D geometric objects. The behavior of 3D map objects is specified by associating events and actions. These objects determine both interactivity and intelligence of 3D maps, and complement the 3D map design.

Dynamic 3D maps facilitate the visual simulation of environmental issues: Due to their effective and efficient visual design capabilities, these maps can be used as interactive tools for presentation, exploration, and manipulation of spatial data as required by a growing number of applications such as spatial decision support systems, vir-

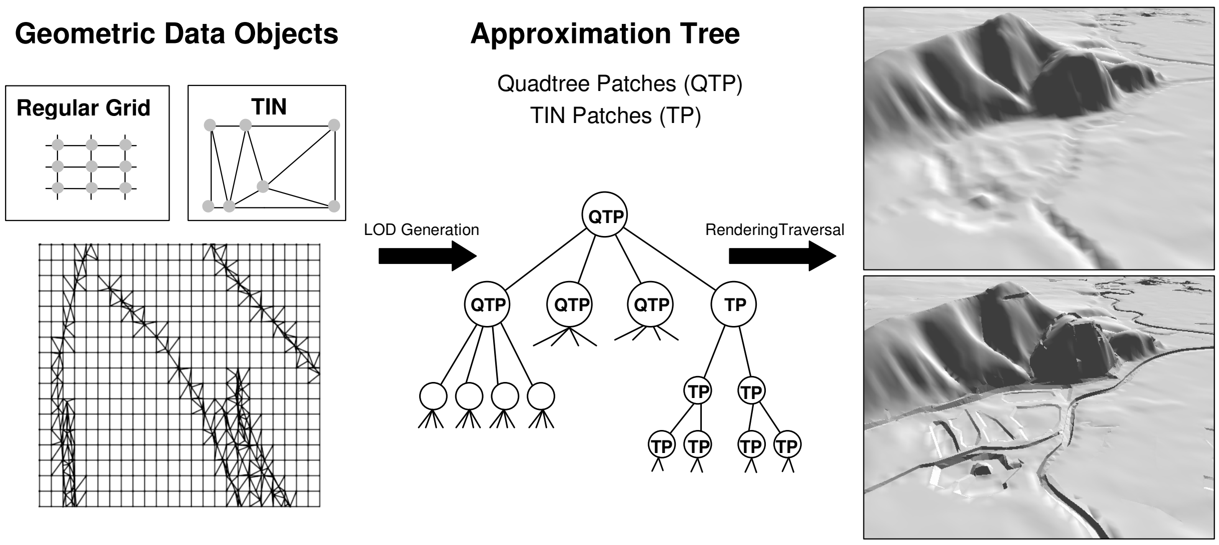

Figure 1. A conceptual view of geometry data objects and approximation tree representations. TIN data and regular grid data form a hybrid mesh. On the right side, the terrain model is visualized without microstructures (top) and with microstructures (bottom).

tual reality applications, real-time GIS, and interactive cartographic environments.

The concepts described in this paper have been implemented in a dynamic 3D map software prototype offering navigation, orientation, exploration, and analysis functionality [2].

The remainder of this paper is structured as follows: Section 2 briefly outlines the hybrid multiresolution terrain model. Section 3 describes how multiresolution texture layers are associated with the level-of-detail terrain geometry. Section 4 focuses on special issues regarding shading textures. Section 5 describes applications of texure layers for visualizing thematic data, and Section 6 discusses how texture layers can be combined. Section 7 outlines the software architecture of our prototype implementation, and Section 8 gives some conclusions.

2. Hybrid Multiresolution Terrain Model

3D maps rely on terrain models as fundamental graphics components. Due to the size of terrain models, multiresolution modeling is a well-known technique to simplify terrain models for the purpose of real-time 3D visualization. We briefly outline a hybrid multiresolution model which integrates different existing, sophisticated multiresolution structures within a single terrain model and operates on geometry data sets having different topological structure. The core component of this approach is the *approximation tree* data structure which constructs the level-of-detail (LOD) surface models for a digital terrain; it has been described in detail in [1].

2.1 The Approximation Tree Data Structure

An approximation tree is specified by a collection of *geometry data objects,* e.g. regular grids or TINs, and a set of tree nodes, called *geometry patches* (see Figure 1). Each geometry patch, i.e., each tree node, represents an approximated terrain surface for a rectangular region at a certain level of detail. A geometry patch provides a *simplification strategy*, i.e., an algorithm for subdividing the terrain domain and a strategy for selecting appropriate points of the geometry data objects according to quality and performance criteria. A simplification strategy operates on the data provided by geometry data objects which include regular grid objects, TIN objects, and arbitrarily triangulated meshes, and calculates or adjusts the internal surface representation. The rendering algorithm selects geometry patches based on screen-space and object-space error criteria [16]. Geometry data objects can be organized in a hierarchical fashion, for instance, a grid data object can have additional TIN objects which refine subregions of the parent grid by microstructures [3]. The hierarchy of geometry data objects is independent from the hierarchy of geometry patches, i.e., the geometry data objects are organized with respect to the application data sources, whereas the approximation tree subdivides the terrain recursively into non-overlapping rectangular terrain patches at different levels of detail according to a given strategy, e.g., quadtrees.

In the past a number of simplification strategies have been proposed for generating multiresolution models for digital terrains (e.g., hierarchical triangulations [9][8][7], progressive meshes [14][15], LOD heightfields [17][19], adaptive meshes [12], restricted quadtree triangulations [21]). While all these techniques have their strengths and weaknesses, the approximation tree concept allows developers to use those strategies which are most appropriate for the specific application data sets by integrating them into a uniform multiresolution model. Therefore the multiresolution model can be customized to the needs of an individual application.

In Figure 1 the illustrated approximation tree represents a hybrid digital terrain model built from a regular grid and additional TINs used to refine the grid partially. Two simplification strategies are used: quadtrees for the regular grid, and a binary partitioning method for the TINs. In Figure 1 (right) the terrain is visualized without and with microstructures. The original regular grid is rendered by approximately 12.000 triangles, and the microstructures consist of 3.000 triangles. The improvement of the visual quality resulting from the combined use of coarse-grained and fine-grained structures is obvious: the contours are sharp and the important morphology is represented correctly.

2.2 Properties of Approximation Trees

The key properties of approximation trees with respect to 3D maps include:

Visual quality. The inclusion of microstructures (e.g., TINs) into coarse grid structures allows us to take advantage of the low memory costs of grids and the exactness of TINs. The goal here is to provide a sharp, morphologically correct image of a terrain. The microstructures improve the human perception of the terrain by introducing sharp edges which facilitate the recognition of forms and shapes by the human visual system.

Preservation of semantics. Since the data sets for coarse and fine structures are not converted or topologically transformed, the identity and semantics of the data sets are preserved in the digital terrain model. The costs for data transformation and duplication are saved, and visualized data objects can be identified directly in the 3D map.

Choice of LOD algorithms. Due to the object-oriented architecture the approximation tree data structure is generic, i.e. it can be customized and extended by application-specific simplification and approximation strategies. For example, constrained Delauny triangulations [23] can be integrated in the approximation tree.

3. Texture Layers

Texture mapping became a fundamental drawing primitive [13], and is excellently supported by low-cost graphics hardware. 3D visualization techniques use more and more texturing as one main mechanism to model and visualize geo-referenced topographic and thematic data. In most 3D maps, 2D textures are projected onto digital terrain models carrying thematic information. In the past LOD mechanisms developed for the hierarchical representation of geometry data did not represent texture data in an

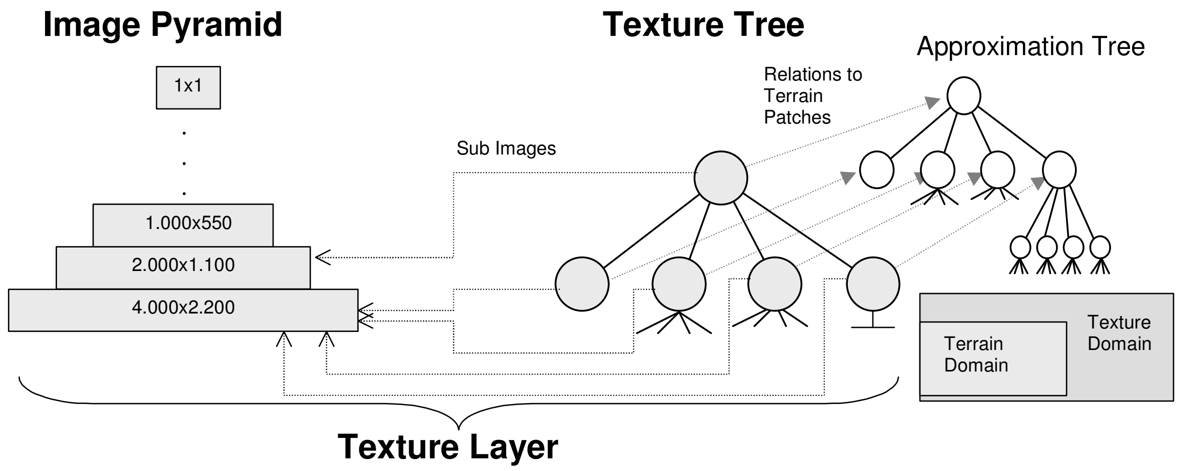

Figure 2. A conceptual view of a texture layer consisting of the image pyramid, the texture tree, and the relations to the approximation tree.

LOD-dependent way. Since map textures are at least as complex as map geometry, we extend the concept of the approximation tree by the texture tree. A texture tree represents a 3D map texture at different levels of detail. A 3D map may have any number of texture layers, that is, any number of texture trees. The approximation tree and its associated texture trees cooperate tightly because the selection of texture patches depends on the selection of geometry patches.

3.1 Image Pyramid and Texture Tree

The construction of a texture layer involves two steps: building an image pyramid and building a texture tree.

The *image pyramid* is derived from the original texture data [24], which might be contained, for example, in a 2D image file. The image pyramid of a texture consists of a sequence of images with decreasing resolution. Each image is created by scaling down the predecessor image by a factor of ½. The first image of the sequence is identical to the original texture data, the last image consists of 1 x 1 pixels. The initial image can have an arbitrary resolution: Cartographic textures, for example, are likely to have a width and height of several ten thousands pixels.

A *texture tree* is specified by a set of tree nodes, called *texture patches*. The texture tree is constructed symmetrically to the approximation tree. Thus, each texture patch is related to exactly one geometry patch, covering or at least overlapping the domain of the geometry patch with respect to its geo-referenced coordinates. Compared to the tree structure of the approximation tree, the texture tree may prune subgraphs if texture patches reference already a subimage of the image pyramid with the highest resolution. Conceptually, a *texture layer* of a 3D map is represented by an image pyramid and a texture tree (see Figure 2).

The rendering algorithm for 3D maps simultaneously traverses the approximation tree and the texture trees for all active texture layers, selecting geometry patches and texture patches according to visual geometric and texture approximation error thresholds.

A texture patch references a sub-image of one of the pyramid's images. It references that image which has highest resolution *and* satisfies the texture constraints imposed by the rendering system. For example, OpenGL [25] defines a maximal texture size and requires that the texture size is a power of 2. The sub-image contains the data actually passed as texture data to the 3D rendering system.

The implementation of texture trees is facilitated by the memory-mapped files provided by the operating system. They permit an application to map its virtual address space directly to a file on disk. Memory-mapped files are useful for manipulating extremely large image files since their creation consumes few physical resources. Then, smaller portions of the file called "views" can be mapped into the address space of the process just before performing I/O. Without memory-mapping, image files not fitting into main memory could not be used as initial images of image pyramids.

If the rendering algorithm decides to use a certain geometry patch and has determined which texture patch to use (in the case of a single texture layer), the texture data is requested. If the texture patch is asked for the first time, the texture patch starts a separate thread loading or calculating the appropriate texture data. As long as the thread is not finished, the texture of the parent texture patch is used instead, which covers the same domain as the child texture patch by definition. Thus, the rendering of a 3D map is progressively refined when texture patches are requested for the first time, but real-time rendering is ensured.

3.2 Multiple Texture Layers

Multiple texture layers are used if multiple thematic data sets have to be projected onto the digital terrain model. For example, land use information and surface temperatures for a given terrain could be visualized by two independent texture layers. Both thematic data sets could be represented by independent texture trees which are used together for display.

Multiple textures are a powerful visual modeling tool: a base texture may carry the shading information at a high resolution thus overcoming the visual artifacts of Gouraud shading, another texture may contain static thematic information, and yet another texture may contain a time-variant texture visualizing flows or movements. The rendering of multiple texture trees is supported efficiently by the multi-texture functionality of the latest (and low-cost) 3D graphics hardware which accelerates the simultaneous projection of two or more textures within one rendering pass.

4. Shading of 3D Maps

3D maps are characterized by both topographic and thematic information. The user perceives and recognizes the morphology of a 3D map mainly by the silhouette and shading of the terrain model. We take that into account in two ways. The approximation tree offers the possibility to visualize topographic detail by including microstructures into the terrain model which leads to a more precise modeling where it is actually necessary. The calculation of a sophisticated shading texture, provided as a separate tex-

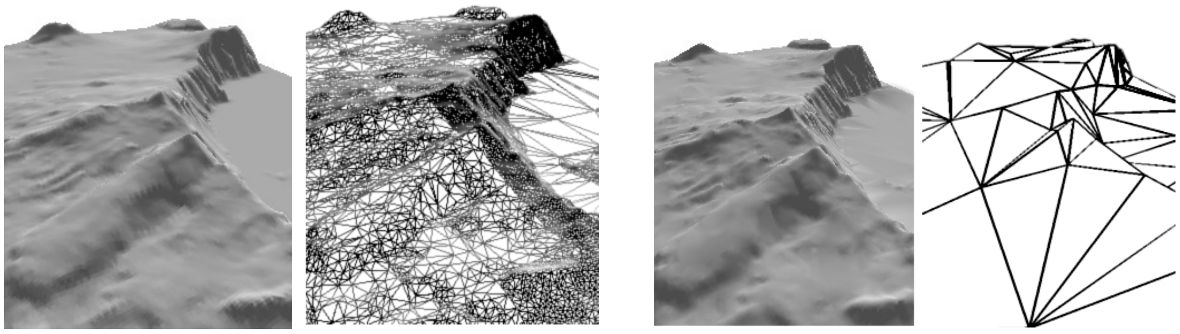

Figure 3. A Gouraud shaded terrain with a high number of triangles (left). A similar terrain with a topographic texture using a considerably lower number of triangles (right).

ture layer, is another possibility to improve the perception of a 3D map.

4.1 Shading Textures

Typically, terrain models are shaded based on Gouraud shading: for each triangle, the shades are calculated at the vertices, and all pixels in the interior of the triangle are colored by interpolating these shades. As a consequence, topographic detail within large triangles is lost, i.e., the topographic quality of a terrain model depends on its geometric resolution because the shading is determined by the triangle vertices. In particular, a wrong visual impression of the terrain's topography results for simplified, low-resolution parts of the terrain.

Strategies for appearance-preserving simplifications take into account that pictorial information is an important factor in perception and can re-introduce visual detail for a geometrically simplified object [5][6]. The strategies rely on the following principle: detailed textures containing appearance information such as shades or surface normals are pre-computed based on the original geometric model and stored separately. Visual detail is introduced into the LOD models by projecting the pre-computed detail textures. Our approach for visualizing topographic and thematic terrain data on maps is based on that principle.

The appearance of topographic features can be preserved in LOD models by pre-calculating a *topographic texture*. Topographic textures allow an application to represent topographic detail without representing that detail geometrically (see Figure 3). Since topographic textures are applied to terrain patches in screen-space, a pixel-precise shading is obtained even for low-resolution terrain parts because the shading quality depends only on the resolution of the topographic texture and not on the geometric resolution.

4.2 Calculation of Shading Textures

The calculation of a shading texture depends on surface properties, surface geometry, topographic features, light sources, and shading rules (e.g., cartographic terrain shading).

In addition, it can take into account topographic features which are classified based on TIN properties [22] in

- zero-dimensional features such as peaks, pits, and saddles describing land form,
- one-dimensional features such as valleys and ridges describing drainages and basins, and
- two-dimensional features such as convex, concave, or flat landform elements.

Topographic textures can be calculated in a preprocessing step by an orthogonal projection of the full-resolution illuminated terrain model into an offscreen or an onscreen framebuffer. The contents of the framebuffer are then used as image data to construct a texture layer. To achieve a resolution higher than the maximal framebuffer size, the topographic texture can be composed of tiles. The full-resolution terrain model can be shaded using the standard OpenGL lighting, an application-specific illumination model (e.g., cartographic hill shading), or it can be based on elevation mapping [10].

If the resolution of the topographic texture turns out to be not sufficient, e.g., in situations where the camera is close to a single polygon of a microstructure, the 3D map calculates on the fly a more precise topographic texture for that region, or even switches to standard Gouraud shading.

Both impressive speed and quality improvements can be achieved using topographic textures on low-cost 3D graphics hardware with accelerated texture mapping because the texture-based approach bypasses the limited geometric processing capabilities on these platforms and saves the costs for per-frame lighting calculations.

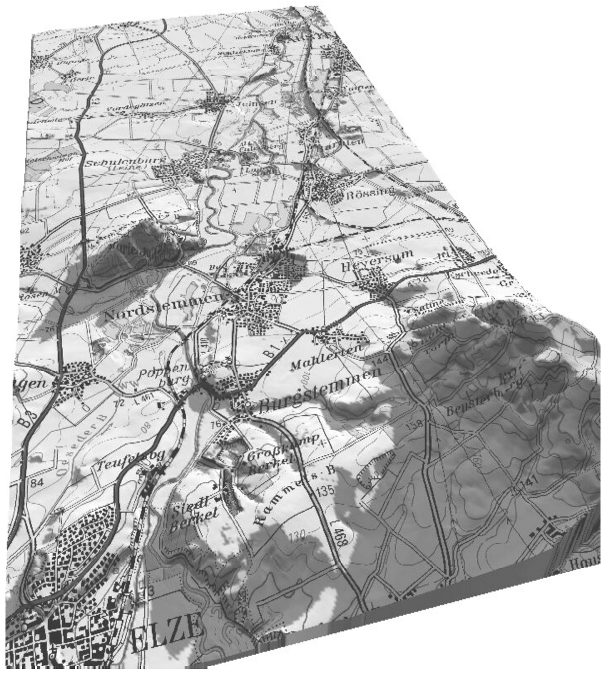

Figure 4. Shading texture with self shadowing, combined with a cartographic texture.

4.3 Examples of Shading Textures

In Figure 4, the shading texture takes into account self shadowing of the terrain model. The self-shadowing is calculated by ray intersection test between light source and full-resolution terrain model. In the example, the shading texture is overlaid with a complex cartographic texture.

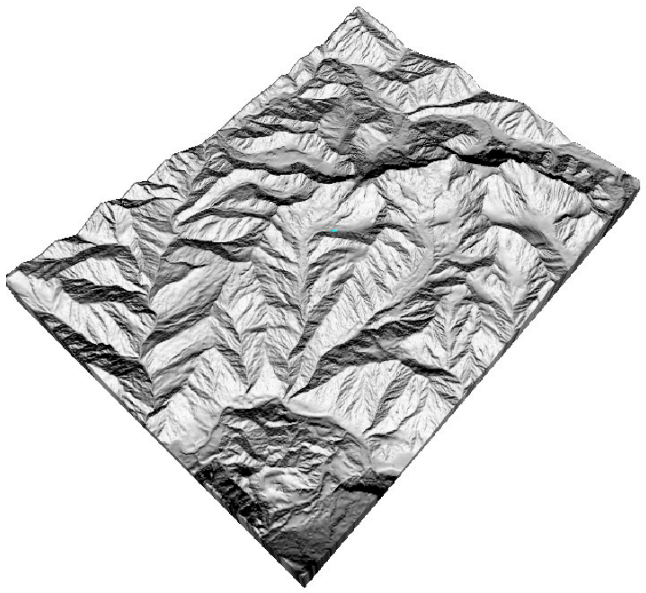

Figure 5. Cartographic shading rules as an example for application-specific shading.

In Figure 5, a 3D map of the Himalaya mountains is shown. The mountains are shaded using a technique which takes into account the relative height of a terrain part, resulting in a more vivid impression of the terrain model.

5. Visualizing Thematic Data

Visualizing thematic data is one of the main purposes of a 3D map. As a key characteristic, thematic data relevant for a 3D map must be geo-referenced and overlap the terrain domain in order to be visualized by a 3D map.

5.1 Thematic Textures

A *thematic texture* is generated by mapping thematic data to a 2D image, and constructing a texture layer based on that image. For rendering of a 3D map, each texture layer can be turned on or off. Frequently, several thematic data sets are visualized simultaneously in 3D maps, for example, road networks and land use information.

Thematic data may already be available as image such as in the case of satellite images and cartographic textures. It may also be generated by applying design rules for thematic data. A simple design rule, for example, could assign color values to the image elements based on a color function. A more elaborated design rule could paint cartographic signatures into the image.

In our approach, design rules have access to the following parameters:

- Image space properties: Size of the image to be generated, resolution of the canvas, and image area of the terrain patch to which the texture should be applied.
- Drawing context: Brush size, brush type, brush color, and paint mode. Used by drawing operations.
- Object space properties: Area of the geometry patch in geo coordinates, surface elements of the geometry patch, image area of the geometry patch to which the texture will be applied.
- Thematic properties: Thematic values available for the area covered by the geometry patch, meta information for the thematic data, for example scalar data type, interpolated data, etc.
- Environmental parameters: Camera distance, position, direction, field-of-view, and light sources (intensity, direction, and light color).

The mapping of thematic data to an image may also depend on the LOD of the terrain patch for which the thematic texture should be constructed. In this case, the texture tree patches reference subimages which may have

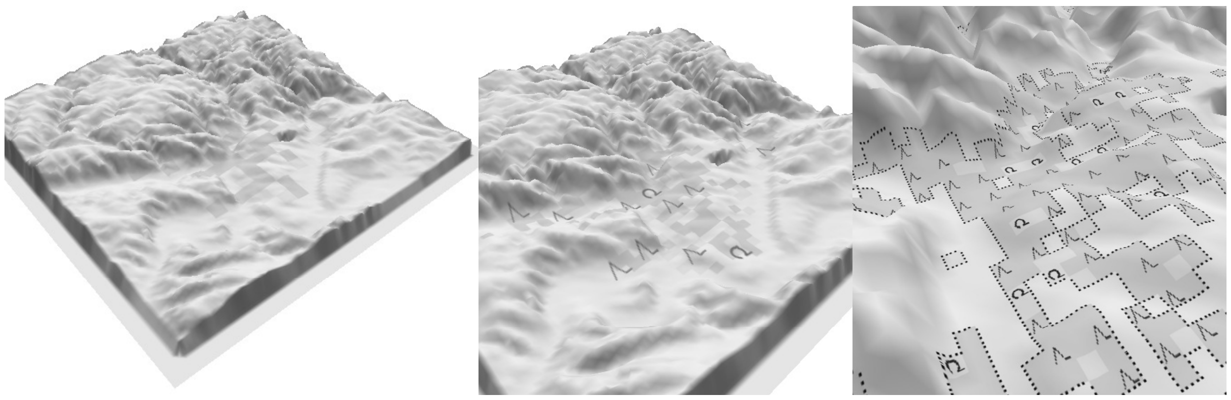

Figure 6. LOD-dependent design rules for forest land use. The exact boundaries of forest areas are visualized if the camera comes close to the terrain. The textures are computed on the fly.

a completely different design. A LOD-dependent 3D map texture is specified by design rules which take into account the LOD. LOD-dependent 3D map textures are used, for example, to implement cartographic generalization schemes (see Figure 6).

5.2 3D Map Objects

In contrast to traditional digital maps, a 3D map may contain 3D objects. For the design of a 3D map, these objects offer an additional way to represent thematic data. Together with texture layers, they are in particular useful to implement map designs which express thematic data in dependency of the level of detail required by 2D textures and 3D glyphs.

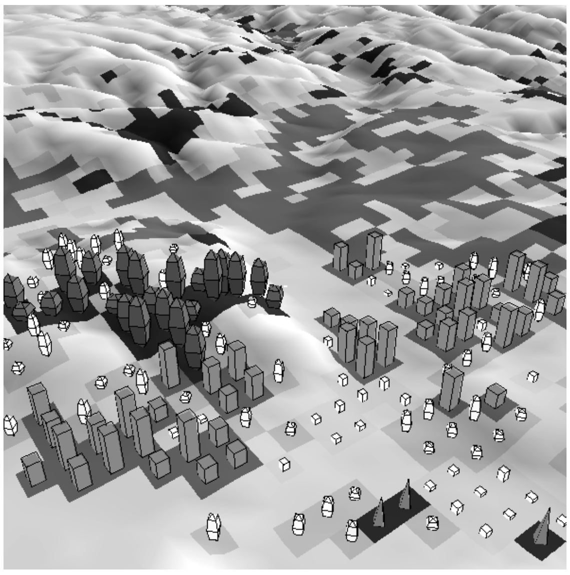

Figure 7. Buildings as 3D map objects, complementary to the texture design of land use information.

In our implementation, design rules are not limited to map thematic data to images. In addition, design rules may decide – at a certain level of detail – to map thematic data to a collection of 3D map objects. We identified the following basic 3D map objects:

- 3D Icons: A 3D icon consists of one, two, or three perpendicular and rectangular textured quads. These quads are rendered as billboards. The visualization of trees by 3D icons is a well-known example.
- 3D Labels: They provide 2D characters as icons, and are rendered as billboards.
- 3D Shapes: A 3D shape provides any type of 3D geometry and must be geo-referenced. Examples are bridges and buildings.

For example, land use information can be visualized by a 2D thematic texture but is complemented in the case of buildings by small 3D icons (see Figure 7) which provide additional information about the building height and type. The buildings, however, are only shown close to the camera.

6. Combining Texture Layers

If a 3D map includes more than one texture layer, it must be specified how these layers are combined. Multiple texture layers can be rendered using multitexturing [25]. Thus, we can provide all texture combination modes available from the underlying 3D rendering system. The operations provided by OpenGL, which is used in our implementation, include blending, adding, subtracting, and multiplying textures. This way, texture layers are kept independent, that is, no costly 2D image operation is necessary in order to combine several texture layers.

Hardware multitexturing supports the rendering of multiple texture layers. Typically two or four independent

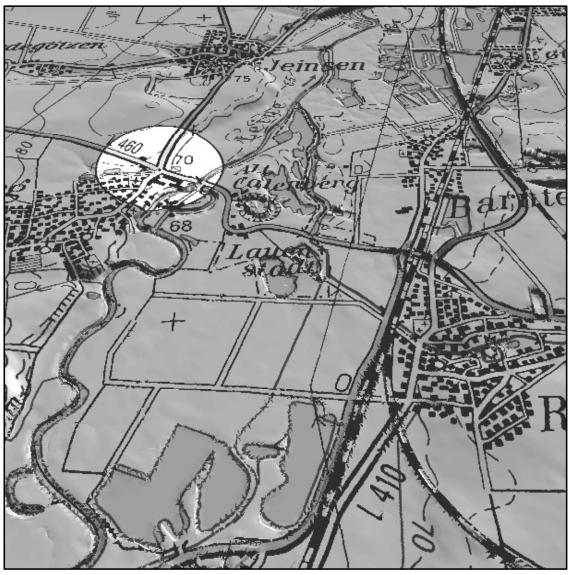
Figure 8. Highlight lens modeled by a transparency texture, a shading texture, and a cartographic texture.

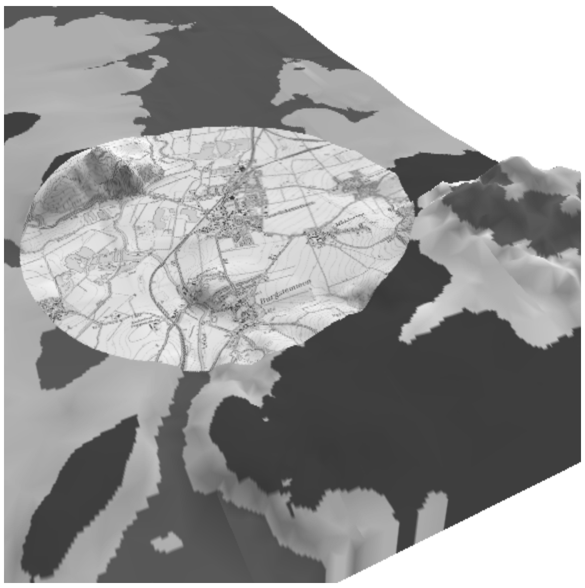
Figure 9. Thematic lens modeled by a weighted interpolation between two thematic textures.

textures can be specified together with their combination modes, and applied simultaneously to the terrain model. During the simultaneous traversal of the approximation tree and all active texture trees, the textures referenced by the texture patches are activated. Note that even if no hardware multitexturing is available, multiple texture layers can be rendered using multiple rendering passes.

6.1 Weighted Interpolation of Textures

The weighted interpolation of two texture layers is frequently needed for 3D maps. The weight is simply specified by an alpha texture, whereby both texture layers are combined in the relation α:(1-α). Both thematic textures and the alpha texture may have different resolution.

The alpha texture may either be pre-computed or specified interactively. Since a typical alpha texture does not require a high resolution, for example the highlight lens in Figure 8 and the "thematic lens" in Figure 9 use a 128x128 texture, the contents of the alpha texture can be calculated during the interaction with the user. In the case of static lens shapes, it suffices to manipulate the texture coordinates of the alpha texture layer, that is, the alpha texture needs not to be recalculated at all.

To optimize real-time rendering, alpha texture layers can be applied to the terrain by disabling the RGB channels of the framebuffer, rendering the terrain only into the alpha channel of the framebuffer (assigning the weight to the framebuffer pixels) where the forthcoming thematic texture has to appear, enabling the RGB channels and alpha testing, and finally rendering the thematic textures in the next passes with the appropriate alpha operators.

6.2 Texture Sequences

Another frequent demand of the 3D map design is to visualize dynamic thematic data. If, for example, time-varying thematic data has been pre-computed as a sequence of texture layers, the data changes can be animated by interpolating between two consecutive texture layers (see Figure 10). The interpolation through a texture sequence uses basically the same mechanism as the weighted interpolation. An animation results if we interpolate between each pair of adjacent textures of the sequence.

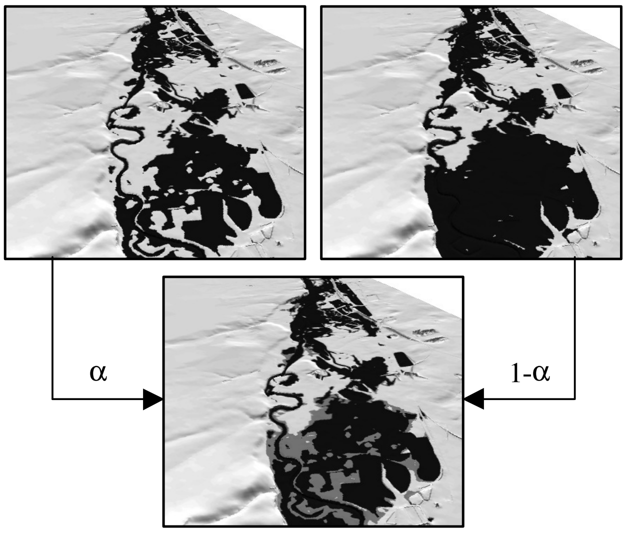

Figure 10. Interpolation through a sequence of textures, animating the flooding in a terrain.

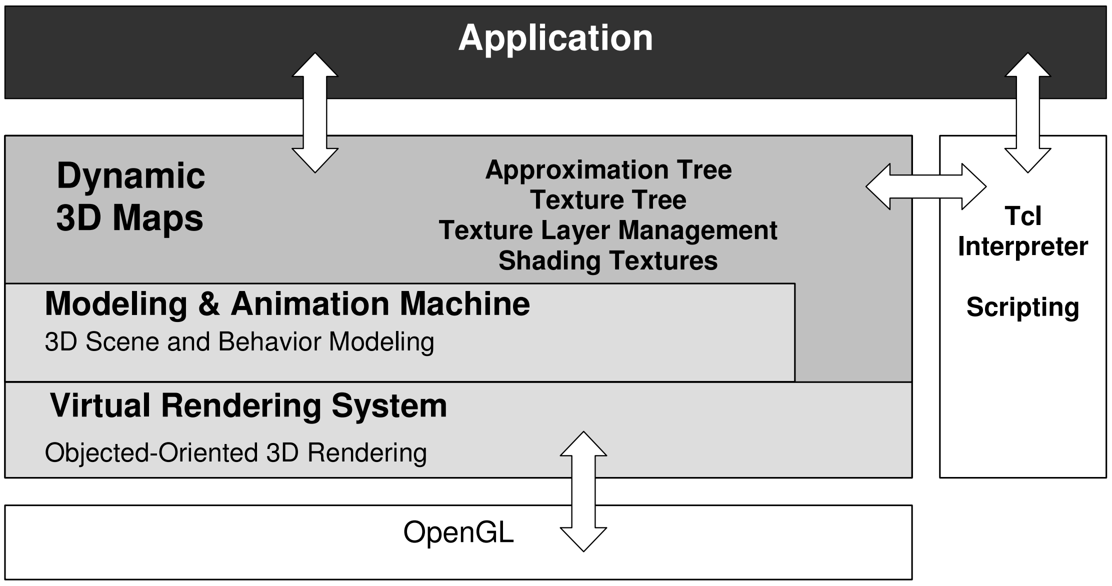

Figure 11. Software architecture of the prototype implementation of the 3D maps.

7. Software Architecture

Figure 11 illustrates the software architecture of the prototype implementation of 3D maps. The implementation is based on the object-oriented modeling and animation toolkit MAM/VRS [11] which consists of two system layers, the Virtual Rendering Systems VRS as a thin object-oriented layer above OpenGL and the Modeling and Animation Machine MAM responsible for 3D scene modeling, 3D interaction, and animation.

A collection of 3D map software components has been developed which interface the data structures and operations for approximation trees and texture trees, shading textures, and texture layer management. 3D maps can be embedded into applications as black-box components. Applications which want to extend the implementation can directly access the C++ classes implementing 3D maps. Design rules can be specified in the scripting language Tcl; the Tcl interpreter is embedded into the C++ application at run-time.

8. Conclusions

3D maps and their texture-based design lead to powerful tools for exploring and analyzing environmental issues and geo data. Their ability to use data sets with different topological structure and preserving the semantics is essential for planing and simulation applications in order to achieve a high visual quality of the terrain model. The tight integration of textures and geometry by the approximation tree and texture tree permits an efficient and flexible application of textures as main tools for designing dynamic 3D map contents. Since topographic textures can be used to encode topographic information, topographic details are preserved which would otherwise be lost due to the LOD models. The texture-based approach for designing 3D maps benefits from recent developments in low-cost graphics hardware which more and more provides fast texturing capabilities. The procedural specifications of thematic textures offers possibilities for easy customization of 3D maps. 3D map objects complement thematic textures and permit to use the third dimension for map design.

Technical details and the prototype implementation are available at the following WWW site: http://www.mamvrs.de

Acknowledgements

We would like to thank Konstantin Baumann and Oliver Kersting for implementing the presented concepts.

References

[1] K. Baumann, J. Döllner, K. Hinrichs, O. Kersting (1999). A Hybrid, Hierarchical Data Structure for Real-Time Terrain Visualization, Proceedings Computer Graphics International '99, 85-92.

[2] G. Buziek, J. Döllner (1999). Concept and Implementation of an Interactive, Cartographic Virtual Reality System. International Cartographic Conference, Ottawa,.

[3] G. Buziek, I. Kruse (1992). The DTM-System TASH in an Interactive Environment. EARSeL Advances in Remove Sensing, 1(3):129-134.

[4] J. Certain, T. Popovic, T. DeRose, D. Duchamp, Salesin, W. Stuetzle (1996). Interactive Multiresolution Surface Viewing. Proceedings of SIGGRAPH '96, 91-98.

[5] J. Cohen, M. Olano, D. Manocha (1998). Appearance-Preserving Simplification. Proceedings of SIGGRAPH '98, 115-122.

[6] P. Cignoni, C. Montani, C. Rocchini, R. Scopigno (1998). A General Method for Preserving Attribute Values on Simplified Meshes. Proceedings Visualization '98, 59-66.

[7] P. Cignono, E. Puppo, R. Scopigno (1995). Representation and Visualization of Terrain Surfaces at Variable Resolution. Proceedings International Symposium on Scientific Visualization '95, World Science, 50-68.

[8] L. DeFloriani (1989). A Pyramidal Data Structure for Triangle-Based Surface Description. IEEE Computer Graphics and Applications, 67-78.

[9] L. DeFloriani, E. Puppo (1995). Hierarchical Triangulation for Multiresolution Surface Description. ACM Transactions on Graphics, 14(4):363-411.

[10] S. Dietrich (2000): Elevation Maps. NVIDIA Corporation, White Paper, http://www.nvidia.com

[11] J. Döllner, K. Hinrichs (1997). Object-oriented 3D Modeling, Animation and Interaction. The Journal of Visualization and Computer Animation, 8(1):33-64.

[12] M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich, M. Mineev-Weinstein (1997). ROAMing Terrain: Real-time Optimally Adapting Meshes, Proceedings Visualization '97, 81-88.

[13] P. Haeberli, M. Segal (1993). Texture Mapping as a Fundamental Drawing Primitive, Proc. of the 4th Eurographics Workshop on Rendering, M. Cohen, C. Puech, F. Sillion (Eds), 259-266.

[14] H. Hoppe (1996). Progressive Meshes. Proceedings of SIGGRAPH '96, 99-108.

[15] H. Hoppe (1997). View-Dependent Refinement of Progressive Meshes. Proceedings of SIGGRAPH '97, 189-198.

[16] R. Klein, G. Liebich, W. Strasser (1996). Mesh Reduction with Error Control. Proceedings Visualization '96, 311-318.

[17] D. Koller, P. Lindstrom, W. Ribarsky, L. F. Hodges, N. Faust, G. Turner (1995). Virtual GIS: A Real-Time 3D Geographic Information System, Proceedings Visualization '95, 94-100.

[18] M. Kraak (1994). Interactive Modelling Environment for Three-dimensional Maps: Functionality and Interface Issues. In: A. M. MacEachren, D. R. Fraser Taylor (Eds.), Visualization in Modern Cartography, Pergamon, 269-285.

[19] O. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, G. A. Turner (1996). Real-time continuous level of detail rendering of height fields. Proceedings SIGGRAPH '96, 109-118.

[20] A. M. MacEachren, D.R. Fraser Taylor (1994). Visualization in Modern Cartography, (Modern Cartography, Vol 2), Pergamon.

[21] R. Pajarola (1998). Large Scale Terrain Visualization Using The Restricted Quadtree Triangulation. Proceedings Visualization '98, 19-26.

[22] M. van Kreveld (1997). Digital Elevation Models and TIN Algorithms. In: M. v. Kreveld, J. Nievergelt, T. Roos, and P. Widmayer (Eds.), Algorithmic Foundations of Geographic Information Systems, Lecture Notes Computer Science, Springer-Verlag, 1340:37-78.

[23] A. Voigtmann, L. Becker, K. Hinrichs (1997). A Hierarchical Model for Multiresolution Surface Reconstruction. Graphical Models and Image Processing, 59:333-348.

[24] L. Williams (1983). Pyramidal parametrics. Computer Graphics (SIGGRAPH '83 Proceedings), 17(3):1-11.

[25] M. Woo, J. Neider, T. Davis (1997). OpenGL Programming Guide, 2nd. Edition, Addison-Wesley.